Teoría de Lenguajes - Recuperatorio del primer parcial

Primer cuatrimestre de 2019

Apagar los celulares.
Hacer cada ejercicio en hojas separadas.
Poner nombre, número de orden y número de página en cada ejercicio.
Justificar todas las respuestas.
El examen es a libro abierto.
Se aprueba con al menos 65 puntos.

1. (25 pts) Dar un autómata finito determinístico para $SUB(INI(L((0^*1)^{++})))$.

2. (25 pts) Determinar si existe alguna expresión regular que denote el lenguaje $L_2 = \{a^m b^n a^r b \ / \ m + n \geq r\}$. De existir, exhibirla. De otro modo, probarlo.

3. (25 pts) Dadas dos cadenas α y β tales que $|\alpha| = |\beta|$ definimos la función $h(\alpha, \beta) = |\{i \mid 1 \leq i \leq |\alpha| \wedge \alpha_i \neq \beta_i\}|$.

   Construir un autómata (finito o de pila, de algún tipo, indicando cuál) que acepte el lenguaje $L_3 = \{\omega \in \{a, b\}^* \mid h(\omega, \omega^r) \leq 1\}$, es decir las cadenas sobre $\{a, b\}^*$ que son "casi palíndromos".

4. (25 pts) Dar una gramática libre de contexto para L_3.

1	2	3	4	9
25	25	25	25	

Ap

HOJA 1

$$L((0^*1)^{++})$$

1) Calcula autómata de L por método de derivada

$$L_0 = (0^*1)^{++}$$

$$\partial_0 L_0 = \partial_0(((0^*1)^+)^+) = \partial_0\left((0^*1)^+((0^*1)^+)^*\right) =$$

$$= \partial_0((0^*1)^+) \cdot ((0^*1)^+)^* \mid \epsilon((0^*1)^+) \cdot \partial_0(((0^*1)^+)^*) =$$

* Calc. aux

$$\partial_0((0^*1)^+) = \partial_0((0^*1)(0^*1)^*) = \partial_0(0^*1)\cdot(0^*1)^* \mid \epsilon(0^*1) \cdot \partial_0((0^*1)^*)$$

$$= [\partial_0(0^*)\cdot 1 \mid \epsilon(0^*)\cdot\partial_0(1)]\cdot(0^*1)^* \mid \underbrace{\phi}_{\lambda\notin L_{0^*1}} \cdot \partial_0((0^*1)^*) =$$

$$= [[\partial_0(0)\cdot 0^*]\cdot 1 \mid \underbrace{\lambda}_{\lambda\in 0^*}\cdot\phi]\cdot(0^*1)^* \mid \phi =$$

$$= [[\lambda\cdot 0^*]\cdot 1\mid\phi]\cdot(0^*1)^* = [0^*\cdot 1]\cdot(0^*1)^* = (0^*1)^+$$

$$\underbrace{\partial_0((0^*1)^+)}\cdot((0^*1)^+)^* \mid \epsilon(\overbrace{(\overbrace{0^*1}^{\lambda\emptyset})^+}^{\lambda\emptyset})\cdot\partial_0(((0^*1)^+)^*) =$$

$$(0^*1)^+\cdot((0^*1)^+)^* \mid \emptyset\cdot\partial_0(((0^*1)^+)^*) =$$

$$= ((0^*1)^+)^+ \mid \emptyset = \boxed{(0^*1)^{++} = L_0}$$

$$\partial_1 L_0 = \partial_1((0^*1)^{++}) = \partial_1((0^*1)^+\cdot((0^*1)^+)^*)$$

$$= \partial_1((0^*1)^+)\cdot((0^*1)^+)^* \mid \epsilon((\underbrace{0^*1}_{\lambda\emptyset})^+)\cdot\partial_1(((0^*1)^+)^*) =$$

$$\lambda\emptyset$$

$$= \partial_1((0^*1)^+)\cdot((0^*1)^+)^* \mid \emptyset\cdot\partial_1(((0^*1)^+)^*) =$$

$$= \partial_1((0^*1)^+)\cdot((0^*1)^+)^* \mid \emptyset = \partial_1((0^*1)^+)\cdot((0^*1)^+)^* =$$

$$= [\partial_1((0^*1)\cdot(0^*1)^*)]\cdot((0^*1)^+)^* = [\partial_1(0^*1)\cdot(0^*1)^* \mid \epsilon(\overbrace{0^*1}^{\lambda\emptyset})\cdot\partial_1((0^*1)^*)]\cdot((0^*1)^+)^* =$$

$$= [[\partial_1(0^*)\cdot 1 \mid \epsilon(\overbrace{0^*}^{\lambda\epsilon})\cdot\partial_1(1)]\cdot(0^*1)^* \mid \emptyset\cdot\partial_1((0^*1)^*)]\cdot((0^*1)^+)^* =$$

$$= [[(\partial_1(0)\cdot 0^*)\cdot 1 \mid \lambda\cdot\lambda]\cdot(0^*1)^* \mid \emptyset]\cdot((0^*1)^+)^* =$$

$$= [[(\emptyset\cdot 0^*)\cdot 1 \mid \lambda]\cdot(0^*1)^*]\cdot((0^*1)^+)^* = [[\emptyset\cdot 1 \mid \lambda]\cdot(0^*1)^*]\cdot((0^*1)^+)^*\cdot$$

$$= [[\emptyset \mid \lambda] . (0^*1)^*] . ((0^*1)^+)^* = [[\lambda] . (0^*1)^*] . ((0^*1)^+)^* =$$

$$= [(0^*1)^*] . ((0^*1)^+)^* = \boxed{(0^*1)^* . ((0^*1)^+)^* = L_1}$$

$$\cancel{\partial_0 L_1 = \partial_0 ((0^*1)^* . ((0^*1)^+)^*) =}$$

Sea $a \Rightarrow a^*(a^+)^* = \underbrace{a^*}_{\text{0 o más } a} \underbrace{(\underbrace{\underbrace{a a^*}_{\text{0 o más } a}}_{\text{1 o más } a})^*}_{\text{0 o más veces [1 o más } a]} = a^*$

$\Rightarrow$ Si $a = 0^*1$ entonces

$$(0^*1)^* . ((0^*1)^+)^* = (0^*1)^*$$

$$\Rightarrow \boxed{L_1 = (0^*1)^*}$$

$$\partial_0 L_1 = \partial_0((0^*1)^*) = \partial_0(0^*1) . (0^*1)^* =$$

$$= [\partial_0(0^*) . 1 \mid \delta(0^*) . \partial_0(1)] . (0^*1)^* =$$

$$= [(\partial_0(0) . 0^*) . 1 \mid \lambda . \emptyset] . (0^*1)^* = [\lambda . 0^* . 1 \mid \emptyset] . (0^*1)^* =$$

$$= [0^* . 1] . (0^*1)^* = \boxed{(0^*1)^+ = L_2}$$

$$\partial_1 L_1 = \partial_1((0^*1)^*) = \partial_1(0^*1).(0^*1)^* =$$

$$= [\partial_1(0^*).1 \mid \epsilon(0^*).\partial_1(1)].(0^*1)^* =$$

$$= [(\partial_1(0).0^*).1 \mid \lambda.\lambda].(0^*1)^* = [(\phi.0^*).1 \mid \lambda].(0^*1)^* =$$

$$= [\phi.1 \mid \lambda].(0^*1)^* = [\phi \mid \lambda].(0^*1)^* = \lambda.(0^*1)^* =$$

$$= \boxed{(0^*1)^* = L_1}$$

Sea $a \Rightarrow (a^+)^+ = a^{++} = (aa^*)^+ = (aa^*)(aa^*)^*$

0 ó + veces

1 ó + veces

1 ó más veces

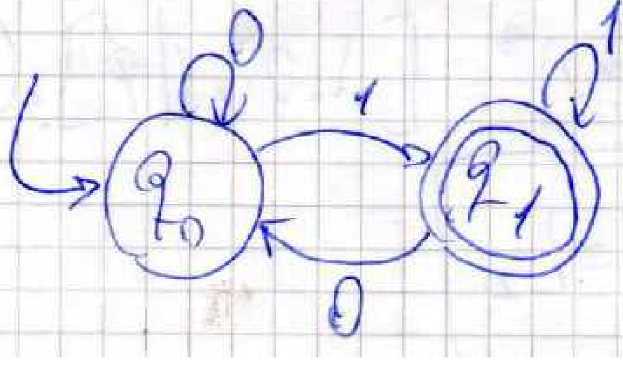
0 ó más veces (1 ó más veces)

1 ó + veces

$a^+ = a^{++}$

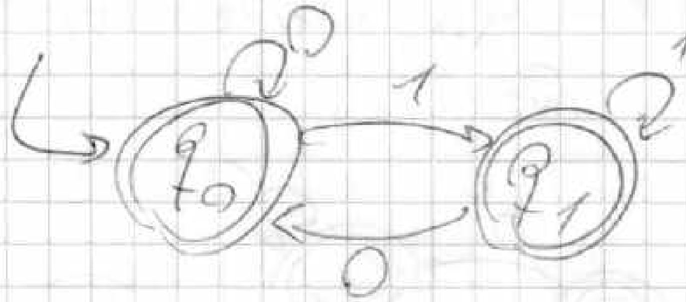
$$\Rightarrow (0^*1)^{++} = (0^*1)^+$$

$$\Rightarrow L_2 = (0^*1)^+ = (0^*1)^{++} = L_0$$



es un autómata de L

Para obtener $\underbrace{\quad}_{\text{autómata de}}$ INI(L) agrego como estado final a todo estado tal que desde ese se pueda llegar a uno final por cualquier cantidad de transiciones

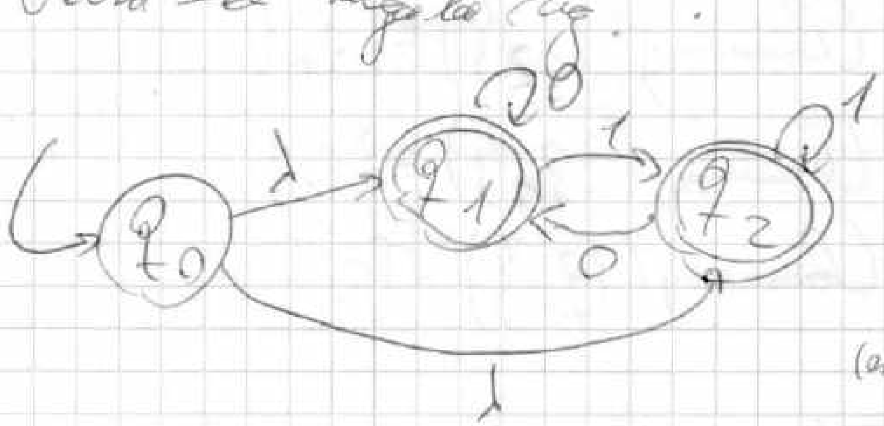


→ autómata de INI(L)

Por último quiero obtener SUB(INI(L))

Para obtener la subcadena de un lenguaje lo 1º que tengo que hacer es finales a todos los estados tales que desde ellos pueda llegar a un estado final, y luego permitir empezar desde cualquiera de esos

Para eso hago lo sig.:

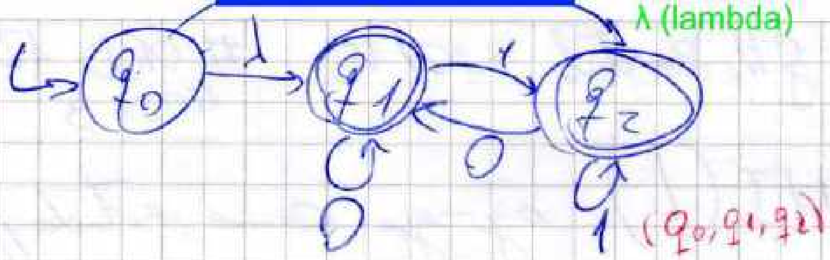


este autómata acepta todas las cadenas, ya se puede concluir que es equivalente a q 0,1

(es más un detalle que una conclusión)

agrego un estado inicial y desde ese pongo transiciones λ

Lo transformo entonces en determinístico

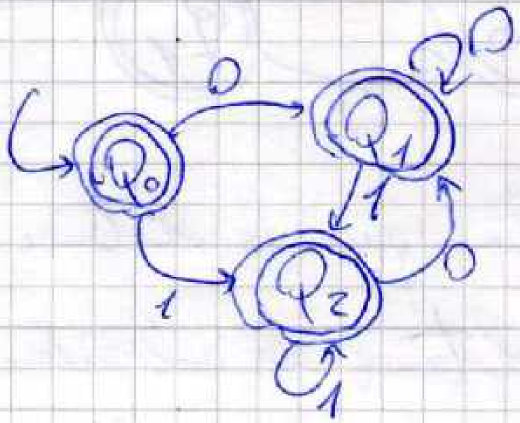


$Q' = \{\{q_1, q_2\},$

$\text{Clausura}_\lambda(\{q_0\}) = \{q_1, q_2\}$

	0	1
x $\{q_1, q_2\}$	$\{q_1\}$	$\{q_2\}$
x $\{q_1\}$	$\{q_1\}$	$\{q_2\}$
x $\{q_2\}$	$\{q_1\}$	$\{q_2\}$

Tomo $Q_0 = \{q_1, q_2\}$ $Q_1 = \{q_1\}$
$Q_2 = \{q_2\}$



*

$\text{mover}(\{q_1, q_2\}, 0) = \text{Clausura}_\lambda(\{q_1\}) = \{q_1\}$

$\text{mover}(\{q_1, q_2\}, 1) = \text{Clausura}_\lambda(\{q_2\}) = \{q_2\}$

$\text{mover}(\{q_1\}, 0) = \text{Clausura}_\lambda(\{q_1\}) = \{q_1\}$

$\text{mover}(\{q_1\}, 1) = \text{Clausura}_\lambda(\{q_2\}) = \{q_2\}$

$\text{mover}(\{q_2\}, 0) = \text{Clausura}_\lambda(\{q_1\}) = \{q_1\}$

$\text{mover}(\{q_2\}, 1) = \text{Clausura}_\lambda(\{q_2\}) = \{q_2\}$

Entonces el autómata pedido es

$\langle \{Q_0, Q_1, Q_2\}, \{0, 1\}, \delta, \{Q_0\}, \{Q_0, Q_1, Q_2\} \rangle$

δ determinada por *

HOJA 4

que reconoce L

2) En la práctica vimos una manera de, dado un AFD obtener un AFND ~~del~~ de L^r, que luego aplicando un algoritmo se puede llevar a autómata finito

El enunciado pregunta por expresión regular!

Entonces si L es regular $\Rightarrow \exists$ AFD que reconoce L

$\Rightarrow \exists$ AFD que reconoce $L^r \Rightarrow L^r$ es regular

Entonces L regular $\Rightarrow L^r$ regular

y su contrarrecíproca

L^r no regular $\Rightarrow L$ no regular, son válidas

Sospecho que L_2 no es regular, intentaré probarlo viendo L_2^r.

Asumo que L_2^r es regular, entonces por lema de pumping

$\exists\, n \,/\, \forall \alpha \in L_2^r \quad |\alpha| \geq n$ vale que

$$\alpha = uvw \text{ con } |uv| \leq n \text{ y } |v| \geq 1$$

$$\text{y } \forall_i \; uv^iw \in L_2^r$$

$$L_2^r = \{ b a^r b^n a^m \,/\, m+n \geq r \}$$

Sea n la constante de pumping $n > 0$

Tomo $\qquad \alpha \in L_2^r$ para $r = n + m$

r = n + m

<= n + m

$b a^{n+m} b^m a^m = \alpha$ su longitud es mayor a

mayor a 'm'

entonces lo puedo descomponer como uvw

Donde $|u| > 0$

$u = b a^k \quad \underbrace{v = a^{j}}_{j \geq 1} \quad w = a^{n+m-k-j} b^m a^m$

$u v^{n+m+1} w$ tiene que pertenecer

$\shortparallel$

$\underbrace{b a^k (a^j)^{n+m+1} a^{n+m-k-j}}_{\downarrow} b^m a^m$

$k + (n+m+1) j + n + m - k - j$

$\tilde{r} = j(n + m + 1 - 1) + n + m$ cantidad de a's

$\tilde{r} = j(n+m) + n + m = (n+m)\underbrace{(j+1)}_{\geq 2}$

$\Rightarrow (n+m)(j+1) > n+m$

$\Rightarrow \tilde{r} > n+m$

$\Rightarrow uvw \notin L_2^r$

HOJA 5

(2) si $|u| = 0$

$v = b a^j$

$w = a^{n+m-j} b^n a^m$

$$|v| \geq 1$$

si $|v| = 1$

$\Downarrow$

$v = b \qquad w = \underbrace{a^{n+m}}_{\geq 1 \to \text{ porque } m>0} b^n a^m$

Entonces tenemos

$$u v^0 w = u w = w = \underbrace{a^{n+m}}_{\geq 1} b^n a^m$$

$\Rightarrow$ empieza con a

$\Rightarrow$ no pertenece a L_2^r

si $|v| > 1$

$v = b a^j \qquad w = a^{n+m-j} b^n a^m$

$$j \geq 1$$

$b^n a^m$

Como $u v^4 w = v^4 w = (b a^j)^4 a^{n+m-j} b^n a$

$= ba^{j}ba^{j}ba^{j}ba^{j} \cdot a^{n+m-j}b^{m}a^{m}$

$\underset{\geq 1}{a^{j}} \; \underset{\geq 1}{a^{j}} \rightarrow$ porque $j \geq 1$

$\Rightarrow$ ~~Empieza~~ no cumple el formato de L_2^r

pues no pueden aparecer más de 2 grupos distintos de "a" (ni de "b") en c/ palabra del lenguaje

$\Rightarrow$ no pertenece a L_2^r

Para cada caso posible de la descomposición uvw de α encontré que $\exists i \,/\, uv^iw \notin L_2^r$

~~Entonces~~

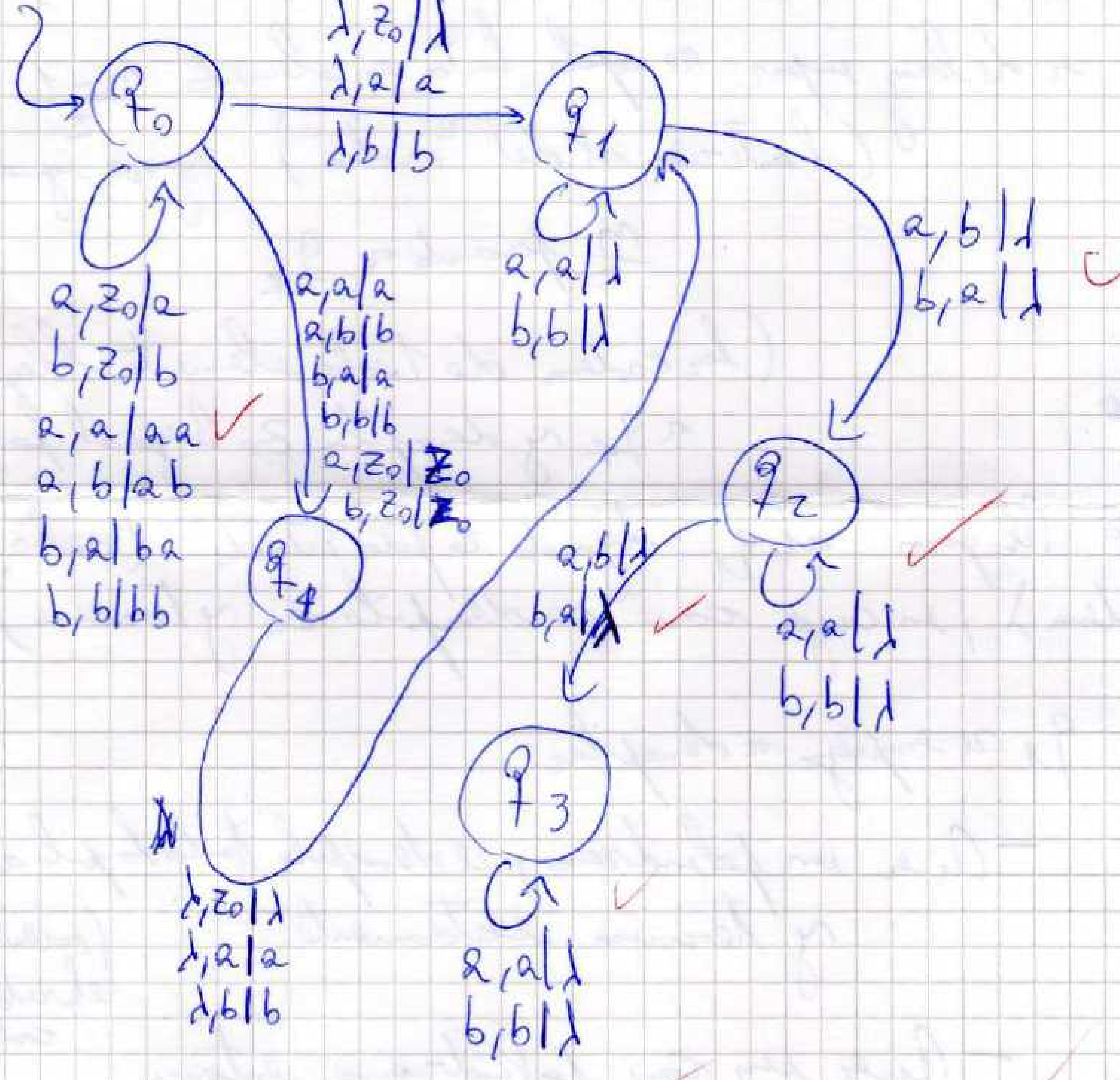
absurdo que viene de suponer L_2^r regular

Entonces L_2^r no es regular

$\Rightarrow \boxed{L_2 \text{ } \underline{\text{NO}} \text{ es regular}}$

3) Hago un autómata de pila

Termina por pila vacía, es no determinista



$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{a, b\}$ $\Gamma = \{a, b, Z_0\}$

q_0 estado inicial

Z_0 símbolo inicial de la pila

Termina por pila vacía

~~Idea: en q_0~~ (y se saca el Z_0)

Idea: en q_0 se apila la primer mitad de la cadena

- Si es ~~par~~ de long par mayor a 0 se apila la primer mitad normalmente
- Si es de long impar se apila hasta en elemento $\frac{n-1}{2}$ (el anterior al del "medio") y se ignora el $\frac{n}{2}$ pasando a q_1

(las cadenas de 1 solo elemento llegan a q_1 y desapilan Z_0 luego al pasar a q_1)

Luego se pasa al q_1 dejando la pila intacta (salvo la cadena λ, en cuyo caso se desapila Z_0 y termina)

En q_1 se empieza a desapilar

- Si es un palíndromo, se desapila toda la pila allí y termina exitosamente (mientras consume el resto de la cadena)
- Si es un casi palíndromo entonces llegará un punto en que el símbolo entrante no coincide con el tope de la pila (esto sucederá dos veces) en cuyo caso se pasa a otro estado, desapilando el elemento que no coincide
- Si sucede que no coinciden más de 2 elementos entonces no reconoce la cadena

HOJA 7

4) $G = \langle \{S, T\}, \{a, b\}, P, S \rangle$

P:

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \lambda \mid aTb \mid bTa$

$T \rightarrow aTa \mid bTb \mid a \mid b \mid \lambda$

Idea:

Con S controlo que se forme un palíndromo, y permito hasta 1 ~~caso de~~ letra que evita que lo sea, ~~cuan~~ cuando sucede esto, T se "encarga" de ~~ter~~ terminar de generar la palabra, ~~evitando que pueda volver a aparecer otro símbolo~~ logrando que todo lo que genere (y esté en el interior de la cadena) SÍ sea un palíndromo, con lo que se termina siendo palíndromo, o casi palíndromo